L25: Entry 8 of 16 File: USPT May 7, 2002

DOCUMENT-IDENTIFIER: US 6385638 B1
TITLE: Processor resource distributor and method

Abstract Text (1):
A device and method for distributing time available on a processor among two or more alternative tasks or threads and scheduling their execution. Each of the tasks specifies multiple levels of time usage on the process under which it can operate, including a best quality of service which uses the most time and a minimal quality of service which uses the least time. The invented system and method guarantees that each task will always be able to run at least its minimal quality of service and includes methods for obtaining optimal quality of service from all of the tasks.

Brief Summary Text (3):
The SMART scheduler, designed at Stanford University, provides support for simultaneous execution of conventional and soft real time applications. Interactive applications get good response. It uses a modified Best Effort scheduler, where in underload all deadlines are met, and in overload, task priorities are used to gracefully degrade performance by denying service to low priority tasks. It uses a notion of virtual time which is advanced for a task as it executes.

Brief Summary Text (8):
The processor is asked to run a set of tasks, such as a set of tasks that provide the appearance of a set of devices, perhaps to a host, or perhaps directly to a user. The host machine may be a DOS box, a computer running Windows, or something else. Alternatively, the processor could be in a set-top box, in which case there is no host. A PC user might be presented with support for audio and video, modem, and graphics devices.

Brief Summary Text (9):
The set of tasks supported in any one system may be static (such as is the case in a DOS box), or it may be dynamic (such as with Windows, or with a set-top box in the presence of Java applets). These tasks must share the processor in real time to present the appearance of simultaneously running devices.

Brief Summary Text (14):
Resource allocation is a step function. Other systems do resource allocation as a smooth function. In other systems, priorities are used in overload to "gracefully degrade" the performance of the running tasks. The problem with this approach is that none of our tasks will "gracefully degrade" if their utilizations are changed slightly. Rather, there are clear steps of quality of service that require corresponding steps in resource allocation. An allocation not at one of these steps results either in wasting resources for a lower level of quality, or missing the deadline for a higher level.

Brief Summary Text (19):
One embodiment of the invention is a method for limiting admissions of tasks to be performed on a processor, each of the tasks having more than one level of use of time, including both a high level which provides a high quality of performance and a low level which provides a low quality of performance. Tasks are admitted for

processing so long as the sum of the lowest use levels for the tasks does not exceed the total time available on the processor. If a task requests processing but the time is unavailable on the processor (the sum of the lowest use levels of the already admitted tasks is too great to admit the new task), the new task is excluded. However, a different task with a lower lowest use level may be admitted if the use level is low enough to fit within the time available.

Detailed Description Text (18):
In the processor resource distributor 1, policy decisions are made by the Resource Manager 5, with reference to a Policy Box 9. The Policy Box is preset by the system designers, but may be modified by the user 8 to match their personal preferences. For instance, by default, video processing should degrade before audio processing, but a deaf user might want to reverse that.

Detailed Description Text (26):
An implication of the fact that we do strict admissions control is that the Scheduler 4 is guaranteed to be asked to schedule only a set of tasks which are known to be schedulable. This means that we can (and do) use a simple scheduling policy like Earliest Deadline First (EDF). EDF has been proven to effectively schedule any set of tasks which is theoretically schedulable, within certain constraints. The constraints include preemptability, and lack of synchronization between the tasks. A task with a smaller period may run several times during a single period for a longer-running task. This requires that the longer running task be preemptable. Scheduling cannot be guaranteed unless the tasks can accept their utilization at any point in a period.--This implies that periodic tasks cannot wait on synchronization within their period. The periodic tasks of a typical set of tasks for the processor resource distributor, controlling video, audio, and communications, meet these requirements.

Detailed Description Text (30):
If a task can only support a single level of quality of service, it would have only one entry in its Resource List. If there are several entries in the table, they are ordered by rate. The entry with the largest rate represents the best performance for the task; the entry with the smallest rate represents the minimum amount of processing that the task could reasonably get by with. Application developers can change the entries in the Resource List with a call to the Resource Manager 5.

Detailed Description Text (38):
When a task enters the system, step 22 or leaves the system, or when a potentially quiescent task changes state, step 25, the Resource Manager must generate a new set of grants for all admitted tasks. At this time, the grant for a task may increase or decrease. As a first step in this process, it determines whether all tasks can get their maximum grant, step 26. If they cannot, the policy box is accessed to get the user's performance preferences, step 27, and the compromised grant set is calculated, step 28. With the new grant set established, the next step is to remove or decrease existing grants with the scheduler to free up resources, step 29, and the final step is to notify the scheduler of new and increased grants, step 30 The task will be informed indirectly because its utilization for the next period will be started with a call to the function associated with the new grant. Because the Resource Manager ensures that the sum of grants does not exceed 100%, the Scheduler 4 need only enforce the grants to be able to use a simple EDF scheme to successfully schedule all tasks. ##EQU1##

Detailed Description Text (87):
The quiescent task may return to the quiescent state (as can any task) with a call to the Resource Manager 5, which again reconfigures the grant set, and suspends the task. An example normal task: MPEG 2 video decoding

Detailed Description Text (88):
MPBG 2 video decoding is an example of a normal task. It has a natural period of 30

milliseconds, which is the frame refresh rate defined by the MPEG standard. In a particular processor where the invention will be used, about 33% of the processor time will be used to accomplish this task.

Detailed Description Text (90):
Different video standards require different combinations of MPEG frames, but a display could contain a random selection of frame types. For example, consider a standard which requires decompressing MPEG frames in the order (IPBBPBBPBBPBBPB) (repeat) From this, one possible time list for an MPEG task would be as shown in Table 4 below:

Detailed Description Text (99):
The method of grant allocation and scheduling works ideally when the time required per period is relatively constant. Some periodic tasks, such as MPEG and audio, fit this model extremely well. There are other tasks, such as 3D, whose processing requirements are dependent not on some fixed value, such as the number of pixels on the screen, but on some highly variable value, such as the number of polygons in a scene. Most 3D tasks have a Polygon Budget which tries to bound this problem, but current Budgets of 5,000-10,000 polygons per frame still leave a performance variation of 2:1.

Detailed Description Text (101):
To address this problem, we have invented the idea of load shifting. Many systems expect a task to do load shedding: perhaps dropping partially processed polygons part-way through the calculations for a frame. In addition to such load shedding, we also allow load shifting which is the controlled transfer of time for one task to another for a limited duration. This allows us to allocate a lesser amount of time to a task like 3D on a regular basis, while still making the extra time available to it on an as-needed basis, all while not damaging the performance of another task.

Detailed Description Text (108):
Aperiodic tasks are handled by a sporadic server. A sporadic server runs periodically with an extremely small utilization, checking a queue for work. If work is available, it contacts the Resource Manager 5 to request a larger grant. The minimum entry in the Resource List for the periodic server is used for the mode where the sporadic server is idle but checking for work to do, as shown by step 50 in FIG. 4. When work arrives, a new Resource List with the same minimum entry, but also with entries requesting increased utilization, is presented to the Resource Manager. If the system is not overloaded, a grant will be made to the sporadic server, which it will use to do the aperiodic work. When the work is complete and the queue empty, the sporadic server makes another call to the Resource Manager with just the minimum entry in its Resource List.

Detailed Description Text (112):
The processor resource distributor provides approximately 500 microsecond granularity. However, there are tasks that need to run more than 1000 (or even more than 2000) times per second, for which this mechanism will not work. Obviously, the tasks running more than 1000 times per second are also using less than 1 millisecond per iteration. These are low latency, high frequency tasks. An example is a DMA interrupt occurring on behalf of the audio command stream, which may interrupt 2800 or even 8000 times per second, but only a few instructions are executed on each iteration. Further, the latency requirement for responding to these events is on the order of a single microsecond.

Detailed Description Text (116):
In the Windows environment and in the set-top box environment, we do this because any task must eventually initiate a request through a driver. Further, the requests are generated by the user in an interactive fashion (at least indirectly). For instance, the user selects "play" from their CD player.

Detailed Description Text (147):
Set top Box--A non-distributed system, not a PC, which probably has a user interface, and which makes available media devices. "Set top" refers to being on top of a TV set.

Detailed Description Paragraph Table (4):
TABLE 4 Possible Resource List for an MPEG Video decoding task: load shedding drops 1/4, 1/3 or 1/5 of the frames. Period Utilization (milli- (milli- seconds) seconds) Rate Function BEST 30 10 10/30 = FullDecompress() 33.3% 120 30 30/120 = Drop_B_in_4 () 25% 90 20 20/90 = Drop_B_in_3() 22.2% MINIMUM 120 20 20/120 = Drop_2B() 16.7%

Other Reference Publication (5):
A Rate-Based Exec;ution Abstraction for Multimedia Computing. In Proceedings of the Fifth International Workshop on Network and Operating Systems Support for Digital Audio and Video, K. Jeffay and D. Bennett. Apr. 1995.

Other Reference Publication (6):
Modular Real-Time Resource Management in the Rialto Operating System. In Proceedings of the Fifth International Workshop on Network and Operating Systems Support for Digital Audio and Video, M.B. Jones, J.S. Barrerra III, P.J. Leach, R.P. Draves.

CLAIMS:

1. A method in a computer system for admitting tasks to be performed on a processor, comprising:

(a) receiving requests for execution on said processor from at least three tasks each having at least a respective high level of use of time on said processor, the respective high use level providing a high quality of performance of the respective task, and at least two of said tasks each having a respective low level of use of time on said processor, the respective low use level providing a low quality of performance of the respective task;

(b) admitting for processing at least two but not all of said tasks, including at least one of said tasks having a respective low use level, the admitted tasks selected such that the sum of their respective low use levels does not exceed the total time available on said processor and such that there is insufficient unused time on said processor to admit any one of the remaining tasks at its respective low use level; and

(c) excluding from admission for processing said remaining tasks.

9. A method in a computer system for adjusting allocation of time on a processor between at least two tasks, comprising:

(a) dividing time on said processor into a series of is periods, commencing execution on said processor of a first task, and allocating to said first task a portion of each of said first periods;

(b) dividing time on said processor into a series of second periods which may or may not be the same as said series of first periods, commencing execution on said processor of a second task at a high level of use during each of said second periods, said high level providing a high quality of performance of said second task, said second task also being capable of executing at a low level of use during said second periods, said low level using less processor time than said high level and providing a lower quality of performance than said high level, and allocating to said second task a portion of each of said second periods; and

(c) if said first task requires more processor time in one of the first periods than the processor time allocated to it, switching said second task to its low level of use and reallocating at least a portion of the time on said processor which is thereby made available to said first task.

13. The method of claim 9 further comprising:

(d) between steps (b) and (c), commencing execution on said processor of a third task executing at a high level of use of time on said processor, said high level providing a high quality of performance of said third task, said third task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level; and,

(e) in step (c), switching said second task to its low level of use rather than switching said third task to its low level of use based on a previously made selection that, in the event that said first task requires more processor time than the processor time allocated to it, said second task will be switched to its low use level rather than said third task.

14. A method in a computer system for adjusting use of time on a processor by a task, comprising:

(a) dividing time on said processor into a series of periods;

(b) commencing execution on said processor of a task executing at a high level of use of time on said processor, said high level providing a high quality of performance of said task, said task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level;

(c) allocating to said task a portion of each of said periods; and

(d) if said task requires more processor time in a period than the processor time allocated to it, switching said task to its low use level.

17. A method in a computer system for adjusting use of time on a processor by a task, comprising:

(a) dividing time on said processor into a series of periods;

(b) commencing execution on said processor of a first task executing at a high level of use of time on said processor, said high level providing a high quality of performance of said first task, said first task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level;

(c) allocating to said first task a portion of each of said periods;

(d) commencing execution on said processor of a second task executing at a high level of use of time on said processor, said high level providing a high quality of performance of said second task, said second task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level; and,

(e) if said first task requires more processor time in a period than

the processor time allocated to it, switching said second task to its low level of

use rather than switching said first task to its low level of use based on a previously made selection that, in the event that said first task requires more processor time than the processor time allocated to it, said second task will be switched to its low use level rather than said first task.

18. A computer readable medium containing a computer program which, when run in a computer system, causes the system to perform the following method for admitting tasks to be performed on a processor:

(a) receiving requests for execution on said processor from at least three tasks each having at least a respective high level of use of time on said processor, the respective high use level providing a high quality of performance of the respective task, and at least two of said tasks each having a respective low level of use of time on said processor, the respective low use level providing a low quality of performance of the respective task;

(b) admitting for processing at least two but not all of said tasks, including at least one of said tasks having a respective low use level, the admitted tasks selected such that the sum of their respective low use levels does not exceed the total time available on said processor and such that there is insufficient unused time on said processor to admit any one of the remaining tasks at its respective low use level; and

(c) excluding from admission for processing said remaining tasks.

26. A computer readable medium containing a computer program which, when run in a computer system, causes the system to perform the following method for adjusting allocation of time on a processor between at least two tasks:

(a) dividing time on said processor into a first series of periods, commencing execution on said processor of a first task, and allocating to said first task a portion of each of said periods of said first series;

(b) dividing time on said processor into a second series of periods which may or may not be the same as said first series of periods, commencing execution on said processor of a second task executing at a high level of use during each of said periods of said second series, said high level providing a high quality of performance, said second task also being capable of executing at a low level of use during said second periods, said low level using less processor time than said high level and providing a lower quality of performance than said high level, and allocating to said second task a portion of each of said periods of said second series; and

(c) if said first task requires more processor time in a period of said first series than the processor time allocated to it, switching said second task to its low level of use and reallocating at least a portion of the time on said processor which is thereby made available to said first task.

30. The computer readable medium of claim 26 wherein the method further comprises:

(d) between steps (b) and (c), commencing execution on said processor of a third task executing at a high level of use of time on said processor, said high level providing a high quality of performance, said third task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level; and,

(e) in step (c), switching said second task to its low level of use rather than switching said third task to its low level of use based on a previously made selection that, in the event that said first task requires more processor time than

the processor time allocated to it, said second task will be switched to its low use level rather than said third task.

31. A computer readable medium containing a computer program which, when run in a computer system, causes the system to perform the following method for adjusting use of time on a processor by a task:

(a) dividing time on said processor into a series of periods;

(b) commencing execution on said processor of a task executing at a high level of use of time on said processor, said high level providing a high quality of performance, said task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level;

(c) allocating to said task a portion of each of said periods; and

(d) if said task requires more processor time in a period than the processor time allocated to it, switching said task to its low use level.

34. A computer readable medium containing a computer program which, when run in a computer system, causes the system to perform the following method for adjusting use of time on a processor by a task:

(a) dividing time on said processor into a series of periods;

(b) commencing execution on said processor of a first task executing at a high level of use of time on said processor, said high level providing a high quality of performance of said first task, said task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level;

(c) allocating to said first task a portion of each of said periods;

(d) commencing execution on said processor of a second task executing at a high level of use of time on said processor, said high level providing a high quality of performance for said second task, said second task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level for said second task; and,

(e) if said first task requires more processor time in a period than the processor time allocated to it, switching said second task to its low level of use rather than switching said first task to its low level of use based on a previously made selection that, in the event that said first task requires more processor time than the processor time allocated to it, said second task will be switched to its low use level rather than said first task.

35. A device in a computer system for admitting tasks to be performed on a processor, comprising:

(a) means for receiving requests for execution on said processor from at least three tasks each having at least a respective high level of use of time on said processor, the respective high use level providing a high quality of performance of the respective task, and at least two of said tasks each having a respective low level of use of time on said processor, the respective low use level providing a low quality of performance of the respective task;

(b) means for admitting for processing at least two but not all of said tasks,

including at least one of said tasks having a respective low use level, the admitted tasks selected such that the sum of their respective low use levels does not exceed the total time available on said processor and such that there is insufficient unused time on said processor to admit any one of the remaining tasks at its respective low use level; and

(c) means for excluding from admission for processing said remaining tasks.

43. A device in a computer system for adjusting allocation of time on a processor between at least two tasks, comprising:

(a) means for dividing time on said processor into a first series of periods, commencing execution on said processor of a first task, and allocating to said first task a portion of each of said periods of said first series;

(b) means for dividing time on said processor into a second series of periods which may or may not be the same as said first series of periods, commencing execution on said processor of a second task executing at a high level of use during each of said second periods, said high level providing a high quality of performance of said second task, said second task also being capable of executing at a low level of use during said periods of said second series, said low level using less processor time than said high level and providing a lower quality of performance than said high level, and allocating to said second task a portion of each of said periods of said second series; and

(c) means for, if said first task requires more processor time in a period of said first series than the processor time allocated to it, switching said second task to its low level of use and reallocating at least a portion of the time on said processor which is thereby made available to said first task.

47. The device of claim 43 further comprising:

(d) means for commencing execution on said processor of a third task executing at a high level of use of time on said processor, said high level providing a high quality of performance for said third task, said third task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level; and,

(e) means for switching said second task to its low level of use rather than switching said third task to its low level of use based on a previously made selection that, in the event that said first task requires more processor time than the processor time allocated to it, said second task will be switched to its low use level rather than said third task.

48. A device in a computer system for adjusting use of time on a processor by a task, comprising:

(a) means for dividing time on said processor into a series of periods;

(b) means for commencing execution on said processor of a task executing at a high level of use of time on said processor, said high level providing a high quality of performance of said task, said task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level;

(c) means for allocating to said task a portion of each of said periods; and

(d) means for, if said task requires more processor time in a period than the processor time allocated to it, switching said task to its low use level.

51. A device in a computer system for adjusting use of time on a processor by a task, comprising:

(a) means for dividing time on said processor into a series of periods;

(b) means for commencing execution on said processor of a first task executing at a high level of use of time on said processor, said high level providing a high quality of performance of said first task, said first task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level;

(c) means for allocating to said first task a portion of each of said periods;

(d) means for commencing execution on said processor of a second task executing at a high level of use of time on said processor, said high level providing a high quality of performance to said second task, said second task also being capable of executing at a low level of use of time on said processor, said low level using less processor time than said high level and providing a lower quality of performance than said high level; and,

(e) means for, if said first task requires more processor time in a period than the processor time allocated to it, switching said second task to its low level of use rather than switching said first task to its low level of use based on a previously made selection that, in the event that said first task requires more processor time than the processor time allocated to it, said second task will be switched to its low use level rather than said first task.